COMPARISON OF MACHINE LEARNING ALGORITHMS IN A HUMAN-COMPUTER HYBRID RECORD LINKAGE SYSTEM

A Thesis

by

MAHIN RAMEZANI FOUKOLAYI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Dilma Da Silva
Co-Chair of Committee,	Hye-Chung Kum
Committee Member,	Mark Fossett
Head of Department,	Scott Schaefer

May 2021

Major Subject: Computer Science

ABSTRACT

Record linkage, often called entity resolution or de-duplication, refers to identifying the same entities across one or more databases. As the amount of data that is generated grows at an exponential rate, it becomes increasingly important to be able to integrate data from several sources to perform richer analysis. In this paper, we present an open source comprehensive end to end hybrid record linkage framework that combines the automatic and manual review process. Using this framework, we train several models based on different machine learning algorithms such as random forests, linear SVM, Radial SVM, and Dense Neural Networks and compare the effectiveness and efficiency of these models for record linkage in different settings. We evaluate model performance based on Recall, F1-score (quality of linkages) and number of uncertain pairs which is the number of pairs that need manual review. We also test our trained models in a new dataset to test how different trained models transfer to a new setting. The RF, linear SVM and radial SVM models transfer much better compared to the DNN. Finally, we study the effect of name2vec (n2v) feature, a letter embedding in names, on model performance. Using n2v results in a smaller manual review set with slightly less F1-score. Overall the SVM models performed best in all experiments.

DEDICATION

To my parents and my sisters

Thank you for being there for me. Your optimism, hope, and love are just what my heart needed;

and to my husband, Amir

for being a constant source of love, support, and encouragement over the years.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Hye-Chung Kum, whose knowledge, experience, understanding, continuous support and patience made this work possible.

I would also like to thank the members of my committee, Dr. Dilma da Silva and Dr. Mark Fossett for finding time out of their busy schedule to reply to my emails, for being ever so kind to show interest in my research and for giving their insightful feedback.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Hye Chung Kum and Professor Dilma da Silva of the Department of Computer Science and Engineering and Professor Mark Fossett of the Department of Sociology. All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work was funded in part by Patient Centered Outcomes Research Institute (PCORI) methods program contract ME1602-34486, and NSF EAGER: Collaborative Research: A Benchmark Data Linkage Repository (DLRep) (SES-1744071).

In addition, the Texas Virtual Data Library (ViDaL) funded by the Texas AM University Research Development Fund which funded the secure computing infrastructure required in this research and access to EHR data was funded in part by the Brown Foundation, NIH NCATS grants UL1 TR000371 and UL1 TR001105. The content is solely the responsibility of the authors and does not necessarily represent the official views of PCORI, NSF, or the Texas AM University. The funding agency had no role in design and conduct of the study; collection, management, analysis, and interpretation of the data; or preparation of the manuscript.

NOMENCLATURE

DL	Damerau–Levenshtein
DNN	Dense Neural Network
EHR	Electronic Health Record
JW	Jaro Winkler
LCS	Longest Common subsequence
ML	Machine Learning
n2v	Name2Vec
NC	North Carolina
RF	Random Forest
RL	Record Linkage
SVM	Support Vector Machine

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION*

As the amount of data that is generated grows at an exponential rate, it becomes increasingly important to be able to integrate data from several sources to perform richer analyses. For example, the research on covid can be accelerated if all fragmented patient data could be integrated. In error-free clean databases with unique identifiers common to all the databases, integrating them can be easily accomplished with simple joins[3]. However, such identifiers are often not available in real world data. In that case, the available fields common to the databases are compared and a decision has to be made on whether the two records refer to the same real world entity or not. This problem of finding data records in heterogeneous databases that refer to the same entities is referred to as *record linkage (RL)* or *entity resolution*. When finding data records for the same entities in one database, this problem is also called *de-duplication* for linking the database to itself.

Automated record linkage methods have been studied extensively in many fields since the problem was first introduced by Newcomb[4]. The best results may be obtained by a hybrid human-computer linkage process that augments the results of automatic algorithms with human judgement[5]. It involves a small team of well trained human experts reviewing potential uncertain pairs generated by algorithms and first making independent decisions then comparing notes on disagreements and coming to consensus [6, 7, 8]. Probabilistic methods and rule-based approaches were the most common automated approaches but machine learning (ML) approaches are rapidly gaining traction and proving to be the preferred automatic linkage methods.

In this research, we present a comprehensive end to end hybrid record linkage framework that combines the manual review and the automated process to achieve both scalability and high quality linkage results. Quality control in any record linkage project is critical because all approaches will result in some level of incorrect matches that will generate erroneous integrated data as well as miss correct matches resulting in a fragmented integrated dataset. We achieve the best of both

*Part of this chapter is reprinted from "Evaluation of Machine Learning Algorithms in a Human-Computer Hybrid Record Linkage System" by M. Ramezani, G. Ilangovan, H-C Kum 2021, AAAI-MAKE Symposium.

worlds by allowing the automated algorithms to resolve majority of the linkages that have a high probability of being either a match or non-match, but also have the option to send ambiguous pairs to human experts for final determination to improve the linkage quality[9]. Hence, the goals of this hybrid record linkage process is to achieve optimum linkage quality, both in terms of no mismatches and no true matches missed, while still minimizing the amount of manual review required to achieve this quality. This research focuses on comparing how well different ML algorithms meet this goal. We also investigate how well different ML models trained on one dataset transfer to other settings within the USA. Determining which ML models transfer better to other settings is important because one of the difficulties to using ML methods on real projects is challenges to building a training set that is comprehensive enough to build good models. In addition, we studied how adding letter embedding[10] in names will effect the performance of these models. In sum, the contributions of this research are:

- A hybrid open source RL framework that can achieve scalability and high quality results
- A comparison of four different RL ML algorithms in meeting the goals of the hybrid system
- A comparison of how well ML RL models trained on one dataset transfer to different settings
- An evaluation on the impact of using letter embedding in names in RL ML algorithms

The rest of this thesis is laid out as follows. In section 2, we briefly review the RL literature on using ML algorithms. Section 3 describes a hybrid record linkage framework and section 4 describes the experimental design of our evaluation. Section 5 and 6 then describe the results from the individual experiments and discuss main insights. Finally, Section 7 presents our conclusions.

2. RELATED WORKS

Linking multiple databases is an important research problem with multiple applications. This problem has been studied for more than six decades [4]. In this chapter, we briefly survey the different machine learning approaches to record linkage.

2.1 Machine Learning Algorithms for Record Linkage

The use of ML algorithms for RL has been studied in different contexts. For instance, a radial basis kernel SVM was used successfully to link genealogy records from 19th century Canada[11]. They designed a record linkage system that incorporates a supervised learning module based on radial SVM for classifying pairs of records as matches and non-matches to link records from the 1871 Canadian census to the 1881 Canadian census.

Kim et al.[12] used random forests for RL in financial entity recognition. Their algorithm works in two steps. First, check if the record pair can be exactly matched after cleaning the entity name and address and then, train a binary Random Forest classifier to decide the linkage. In another study, Treeratpituk et al.[13] demonstrated the efficiency of random forests for disambiguation. They described an algorithm for pairwise disambiguation of author names based on a machine learning classification algorithm, random forests. Different authors may share the same names, either as full names or as initials and last names and the user would like the digital library to differentiate among these authors.

With the recent re-emergence of neural networks, a lot of research shows the potential for neural networks in entity resolution. Pixton et al.[14] used structured neural networks for pedigree-based record linkage. Based on their study, structured neural networks exhibit similar properties to fully connected networks, but their underlying structure makes them more amenable to interpretation. Wilson[15] also demonstrated a huge improvement in accuracy through the use of neural networks, compared to traditional probabilistic record linkage on a large genealogical dataset.

2.2 Comparing different Machine Learning algorithms

There are some studies that compared the performance of different ML algorithms in different settings. In this section, we discuss the approach and results of three different studies.

2.2.1 Feigenbaum Study

Feigenbaum[16] compared the performance of ML algorithms (SVM, random forest), logistic regression, and other heuristic approaches on US census dataset. The data he used contained these variables:

- first name (including middle initial if available)
- last name
- year of birth

He used a supervised learning procedure, teaching an algorithm to discriminate between correct and incorrect matches based on training data generated by the researcher. For evaluation, he considered two different factors and defined them as follow[16]:

1. Efficiency: A high share of the records to be searched for is found and matched. True Positive Rate: $TPR = TP/(TP + FN)$
2. Accuracy: A high share of the records matched are true matches and not false positives. Positive Predictive Value: $PPV = TP/(TP + FP)$

His results (table 2.1 and table 2.2) showed that SVM did slightly better than RF in both true positive rate (TPR) and positive predictive value (PPV).

Model	TPR (Training data)	TPR (Test data)
RF	0.973	0.756
SVM	0.756	0.827

Table 2.1: Efficiency of ML algorithms for Feigenbaum Study

Model	PPV (Training data)	PPV (Test data)
RF	0.947	0.868
SVM	0.904	0.891

Table 2.2: Accuracy of ML algorithms for Feigenbaum Study.

2.2.2 Ilangovan Study

Ilangovan[1] studied the effectiveness and efficiency of different ML algorithms (SVM, Random Forest, and neural networks) in a controlled experiment with different levels of heterogeneity in data and size of the training dataset. He used these variables from North Carolina voter registry dataset:

- first name
- last name
- date of birth
- gender
- race
- a unique ID field common to both databases is used for labeling

And added different type of errors to generate perturbed data. For example:

- duplicates
- Twins
- Suffixes
- day-month swaps

- Nick names
- first-last name swaps
- Last name change due to marriages
- typos like insert, delete, transpose, replace on names and dates

For the evaluation, he used two different factors:

1. F1-score
2. the percentage of manual review

He found that RF and SVM performed very well both in terms of traditional metrics like F1 score as well as manual review set size for error rates from 0% to 60%. Figure 2.1 shows his results.

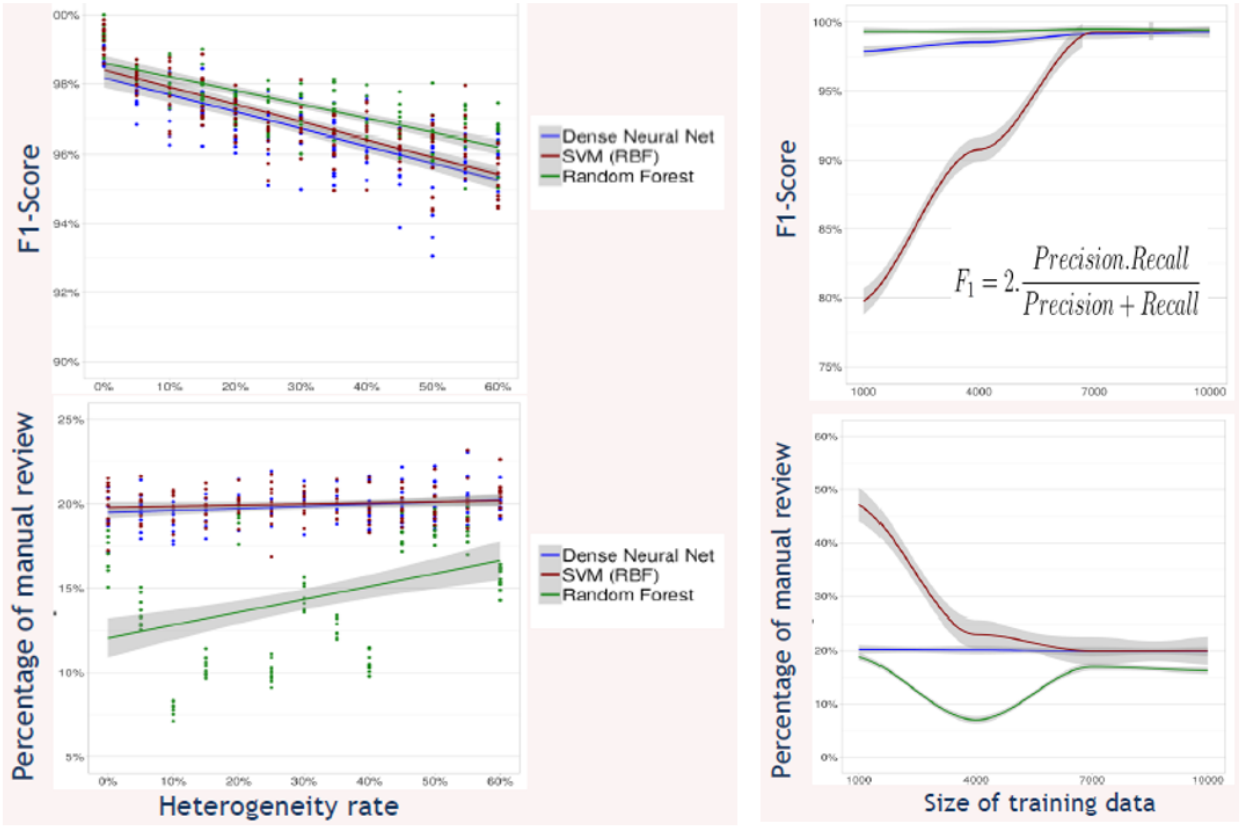

Figure 2.1: F1-score and Percentage of manual review by heterogeneity rate for Ilangovan study. Reprinted from [1].

2.2.3 Kaur Study

In [17] the performance of SVM and Random Forest was compared to investigate the upper bound achieved in the linkage rate and the conditions required to achieve the rate. The results of this study illustrated that the RF produced high quality results at threshold value >= 0.85 while for the cases where quantity is the main concern SVM with a lower threshold is recommended. Using the Canadian census 1871 and 1881, Kaur designed two experiments. In the first experiment, he used these variables:

- Given Name
- Last Name

- Age
- Sex
- Marital Status
- Birthplace

And, in the second experiment, he added six more variables to what he had for the first experiment:

- Province
- Origin
- Religion
- District Number
- Sub District Number
- First Name Code

For the evaluation, he used False Positive rate which is calculated as: $FPR = FP/(FP+TP)$. Table 2.3 and 2.4 show the result of his experiments. Note that SVM_P1 and RF_P1 refer to the support vector machine and the random forest systems implemented using the limited attributes (the first experiment) while SVM_P2 and RF_P2 refer to the systems that incorporated the additional attributes (second experiment).

Model	TP	FP	FPR
SVM_{P1}	3654	230	5.92%
RF_{P1}	3368	256	7.06%
SVM_{P2}	3759	238	5.95%
RF_{P2}	4950	100	1.98%

Table 2.3: False Positive Rate for Kaur Study.

Model	Total Links	Multi Links	Single Links	Linkage Rate
SVM_{P1}	6,781,399	6,279,657	501,742	14.47%
RF_{P1}	15,018,198	14,496,386	521,812	15.05%
SVM_{P2}	2,371,036	1,824,171	546,865	15.77%
RF_{P2}	1,423,554	701,313	722,241	20.83%

Table 2.4: False Positive Rate for Kaur Study.

2.2.4 Comparison of the three study

In this subsection, we are summarizing all three studies in one table (Table 2.5).

Study	Feigenbaum	Ilangovan	Kaur
ML algorithms	RF SVM	RF SVM NN	RF SVM
Variables	first name last name year of birth	first name last name date of birth gender race	Given Name Last Name Age Sex Marital Status Birthplace Province Origin Religion District Number Sub District Number First Name Code
Evaluation	TPR PPV	$F1_{score}$ # of manual review	FPR
Best algorithm	SVM	RF	RF for high quality result SVM where quantity is the main concern

Table 2.5: Summary of three studies.

3. HYBRID RECORD LINKAGE *

In the health sector, incorrectly linking records that belong to different patients with similar identifiers such as twins or family members may lead to serious harm due to incorrect health information (e.g., medication allergies). Thus, often algorithms are tuned to minimize false matches which inevitably increases the rate of missing true matches leaving the health records fragmented. This also leads to its own problems (e.g., incomplete medical history). More problematic is that the unlinked true matches are often biased because there are more issues with identifying information in lower socioeconomic populations such as ethnic names[18]. On the other hand, manual RL methods may be prohibitively time-consuming. One solution is to use a hybrid record linkage framework which combines the automated process and the manual process (see Figure 3.1). First, the automated algorithm will handle the records that have high probability of either a match or unmatch, which for most applications is majority of the data, and then a human expert will resolve those remaining records that the algorithms were uncertain on. Thus, in automated record linkage algorithms one threshold is used to divide the data into two classes (match, unmatch), while hybrid methods use two thresholds to form three classes (match, uncertain, unmatch).

An automated record linkage method contains three main steps:

1. Pair generation
2. Feature extraction
3. Machine Learning algorithms

In the following sections, we are going to discuss these steps in detail.

*Part of this chapter is reprinted from "Evaluation of Machine Learning Algorithms in a Human-Computer Hybrid Record Linkage System" by M. Ramezani, G. Ilangovan, H-C Kum 2021, AAAI-MAKE Symposium. © 2021 Copyright for this paper by its authors.

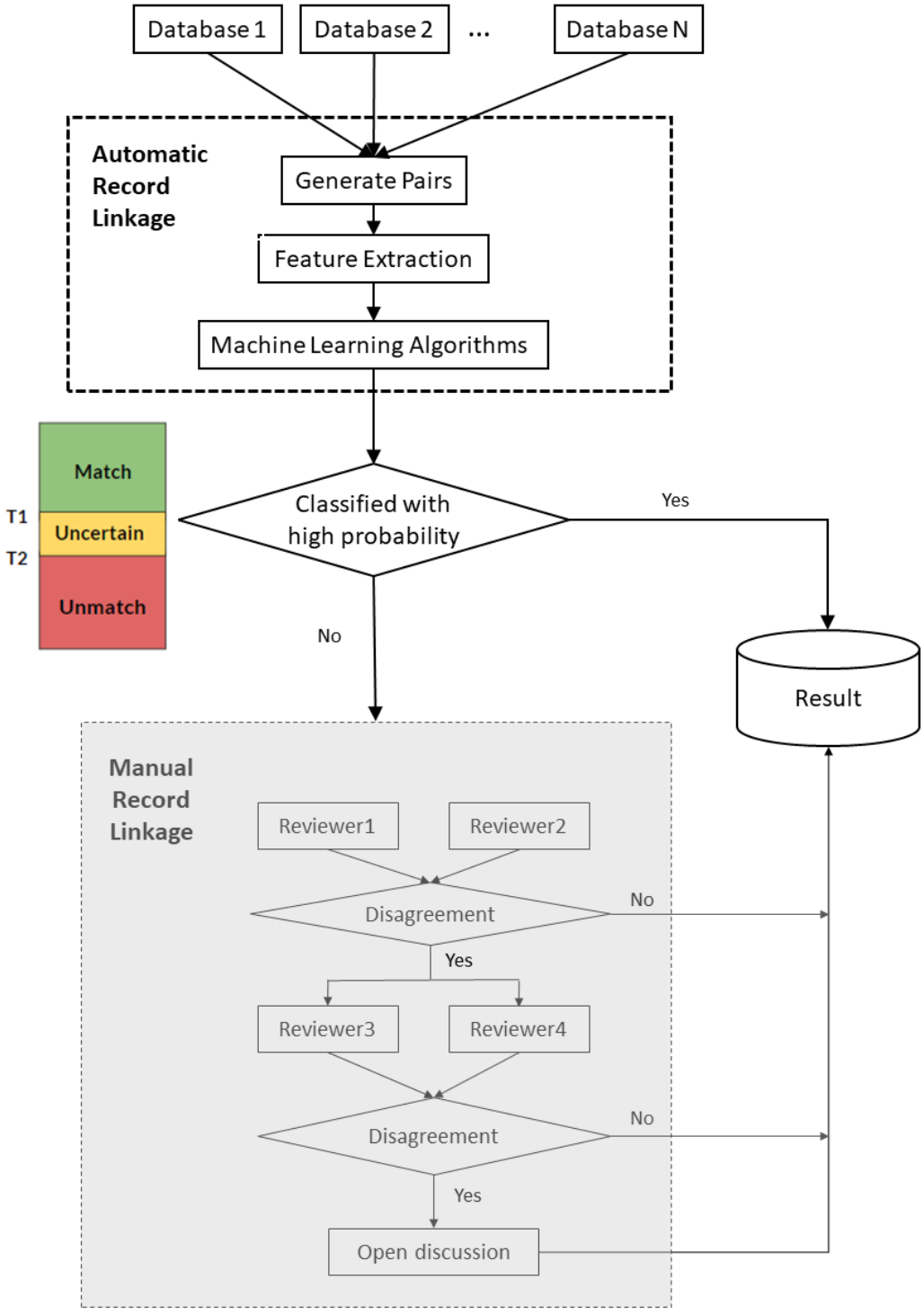

Figure 3.1: Hybrid Record Linkage framework. Reprinted from [2]

3.1 Pair Generation

The first step in the hybrid RL process is the creation of pairs from one or more databases for potential matches such as those that share some common identifier. Often referred to as *blocking*, the idea is to use the identifier field just to generate candidate potential pairs to reduce computation (Figure 3.2).

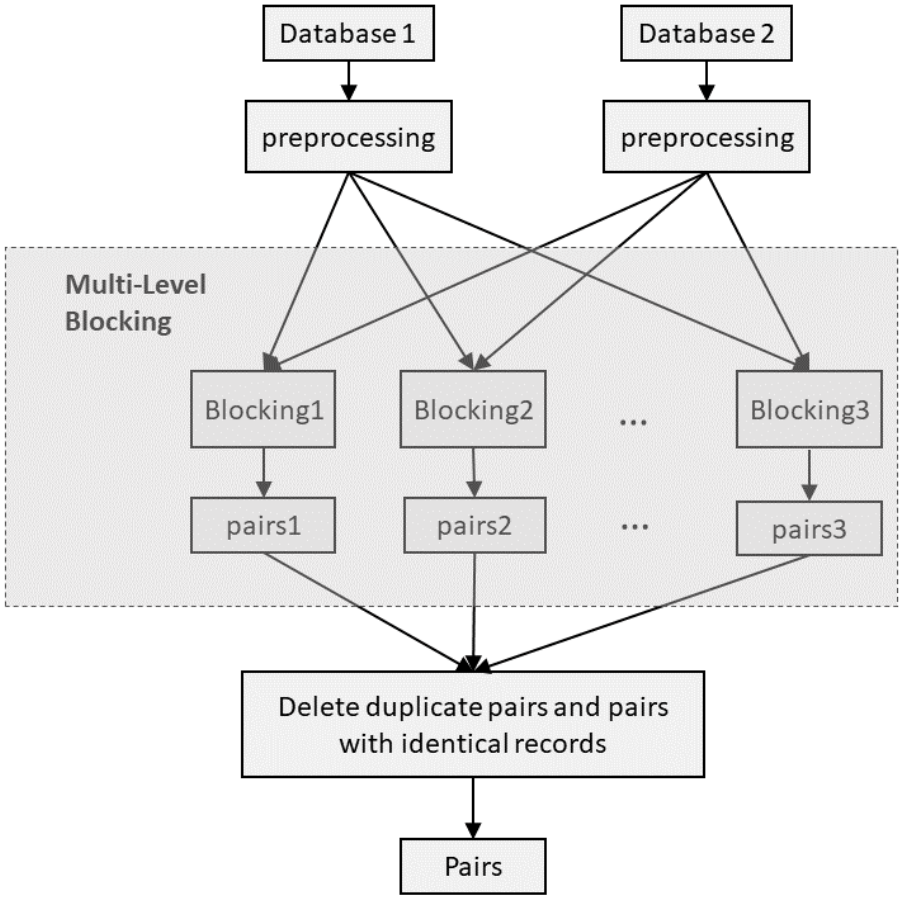

Figure 3.2: Generating pairs from two datasets

Once the pairs are generated, features would be extracted from each pair and fed into the ML models. In this study, we generated the pairs files by blocking on appropriate fields. For example, first name and last name (Table 3.1), first name and date of birth (Table 3.2), last name and date of

birth (Table 3.3), etc.

fname	lname	DOB	sex	race
John	smith	1989-07-28	M	W
John	smith	1998-05-28	M	B

Table 3.1: Blocked on first name and last name.

fname	lname	DOB	sex	race
John	smith	1989-07-28	M	W
John	William	1989-07-28	M	W

Table 3.2: Blocked on first name and date of birth.

fname	lname	DOB	sex	race
John	smith	1989-07-28	M	W
Lucy	smith	1989-07-28	F	W

Table 3.3: Blocked on last name and date of birth.

3.2 Feature Engineering

After generating the pairs, we need to extract some useful features to feed them to ML algorithms. In this section, we first discuss some similarity distances that are used to extract features from the pairs and then we introduce the features that are extracted for ML algorithms.

3.2.1 Similarity Distances

The Distance between two strings s_1 and s_2 ranges from 0 to 1 where 0 means the strings are equal and 1 means no similarity between two strings.

3.2.1.1 Jaro Winkler (JW) distance

The Jaro Similarity[19] of two strings s_1 and s_2 is given by:

$$sim_j = \begin{cases} 0 & m = 0 \\ \frac{1}{3}(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}) & otherwise \end{cases} \tag{3.1}$$

Where $|s_i|$ is the length of the string s_i, m is the number of matching characters (Two characters from two string s_1 and s_2 are considered matching only if they are the same and not farther than $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$ characters apart.), and t is half the number of transpositions.

Jaro–Winkler similarity uses a prefix scale p which gives more favorable ratings to strings that match from the beginning for a set prefix length l. The maximum value for l is 4 and the standard value for p is 0.1 [20]. Thus, the JW similarity between s_1 and s_2 is:

$$sim_w = sim_j + lp(1 - sim_j) \tag{3.2}$$

and the Jaro–Winkler distance is defined as:

$$d_w = 1 - sim_w \tag{3.3}$$

3.2.1.2 Damerau–Levenshtein (DL) distance

Damerau–Levenshtein distance[21] between two strings is the minimum number of operations (consisting of insertions, deletions, or substitutions of a single character, or transposition of two adjacent characters) required to change one word into the other. The Damerau–Levenshtein distance between s_1 and s_2 is given by:

$$d_{s_1,s_2}(i,j) = min \begin{cases} 0 & i = j = 0 \\ d_{s_1,s_2}(i-1,j) + 1 & i > 0 \\ d_{s_1,s_2}(i,j-1) + 1 & j > 0 \\ d_{s_1,s_2}(i-1,j-1) + 1_{(s_1[i] \neq s_2[j])} & i,j > 0 \\ d_{s_1,s_2}(i-2,j-2) + 1 & i,j > 1 \ and \ s_1[i] = s_2[j-1] \ and \\ & s_1[i-1] = s_2[j] \end{cases} \tag{3.4}$$

where $1_{(s_1[i] \neq s_2[j])}$ is equal to 0 when $s_1[i] = s_2[j]$ and equal to 1 otherwise.

- $d_{s_1,s_2}(i-1,j) + 1$ corresponds to deletion from s_1 to s_2.
- $d_{s_1,s_2}(i,j-1) + 1$ corresponds to insertion from s_1 to s_2.
- $d_{s_1,s_2}(i-1,j-1) + 1_{(s_1[i] \neq s_2[j])}$ corresponds to a match or mismatch.
- $d_{s_1,s_2}(i-2,j-2) + 1$ corresponds to a transposition between two successive symbols.

For example, consider two strings "Kathryni" and "Catherine". The DL distance between these strings is 5 since 5 edits are required to change one into the other:

$$Kathryni \xrightarrow{\text{substitution of 'K' to 'C'}} Cathryni$$

$$Cathryni \xrightarrow{\text{insertion of 'e'}} Catheryni$$

$$Catheryni \xrightarrow{\text{deletion of 'y'}} Catherni$$

$$Catherni \xrightarrow{\text{transposition of 'ni' to 'in'}} Catherin$$

$$Catherin \xrightarrow{\text{insertion of 'e'}} Catherine$$

3.2.1.3 Longest Common subsequence (LCS) distance

Longest common subsequence (LCS) distance is an edit distance similar to DL distance with insertion and deletion as the only two edit operations, both at unit cost[22]. Thus the LCS distance for the previous example is 7.

$$
\begin{aligned}
Kathryni &\xrightarrow{\text{deletion of 'K'}} athryni \\
athryni &\xrightarrow{\text{insertion of 'C'}} Cathryni \\
Cathryni &\xrightarrow{\text{insertion of 'e'}} Catheryni \\
Catheryni &\xrightarrow{\text{deletion of 'y'}} Catherni \\
Catherni &\xrightarrow{\text{insertion of 'i'}} Catherini \\
Catherini &\xrightarrow{\text{deletion of 'i'}} Catherin \\
Catherin &\xrightarrow{\text{insertion of 'e'}} Catherine
\end{aligned}
$$

3.2.1.4 Soundex distance

Soundex[23] is a phonetic coding algorithm for indexing names as they are pronounced in English. The goal is for homophones to be encoded to the same representation so that they can be matched despite minor differences in spelling[24]. If the Soundex coding for s_1 and s_2 are equal, the distance between $s1$, s_2 is 0 otherwise 1. The Soundex code for a name consists of the first letter of the name and 3 digits based on table 3.4. Furthermore, there are some additional rules * such as:

- Disregard the letters A, E, I, O, U, H, W, and Y.
- If there are any double letters in the name, consider only the first one. Example: Gutierrez is coded G-362 (G, 3 for the T, 6 for the first R, second R ignored, 2 for the Z).

*https://www.archives.gov/research/census/soundex: :text=Basic

- If there are different letters side-by-side that have the same number in the Soundex coding guide, consider one of them. Example: Pfister is coded as P-236 (P, F ignored, 2 for the S, 3 for the T, 6 for the R).

- If a vowel (A, E, I, O, U) separates two consonants with the same soundex code, the consonant to the right of the vowel is coded. Example: Tymczak is coded as T-522 (T, 5 for the M, 2 for the C, Z ignored (see "Side-by-Side" rule above), 2 for the K). Since the vowel "A" separates the Z and K, the K is coded.

- If "H" or "W" separate two consonants that have the same soundex code, the consonant to the right of the vowel is not coded. Example: Ashcraft is coded A-261 (A, 2 for the S, C ignored, 6 for the R, 1 for the F). It is not coded A-226.

Represents the Letters	Number
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6

Table 3.4: Soundex Coding Guide.

3.2.1.5 Name2Vec (n2v) distance

Name2Vec is a name-embedding using Doc2Vec methodology, where each name is a document and each letter of the name is considered a word. For this study, we trained two separate models using names from two public US datasets. Using code from[10] and all names extracted from the

North Carolina Voter Registration data[†] and ONC Patient Matching Algorithm Challenge data[‡], we trained separate models for first name and last name. Using these models, we can embed names into a vector space and then calculate the cosine distance between the names. The cosine distance for the matching name pairs should be skewed toward 0 while the random name pairs should be around some high value. All cosine scores are in the range [0, 2][10] due to the fact that some of the vectors could have negative values as well.

3.2.2 Features Extraction

3.2.2.1 Name Features

For each pair of first and last names, we calculated the Jaro Winkler (JW) distance, Damerau–Levenshtein (dl) distance, Longest Common Substitution (LCS) distance, Soundex distance, and Name2Vec (n2v) distance. We also created a boolean feature to detect when the first name and last name were swapped. Finally, the normalized frequency of the first and last names in their respective databases was also added to capture how rare the name is. Figure 3.3 shows the features for first and last names.

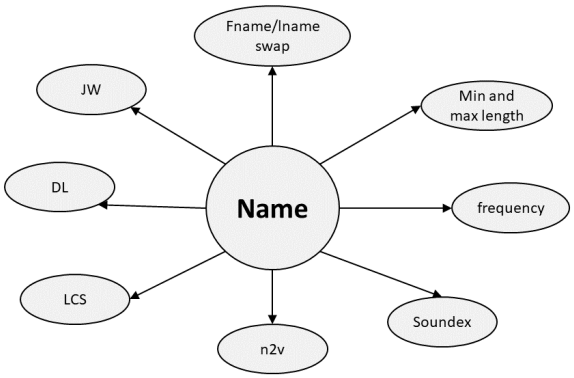

Figure 3.3: Name features

[†]https://www.ncsbe.gov/results-data/voter-registration-data
[‡]https://linkagelibrary.icpsr.umich.edu/linkagelibrary/project/111962

3.2.2.2 Date Features

For each pair of dates of birth, we calculated the Damerau-Levenshtein distance, the Damerau-Levenshtein distance for the year, month, and day components individually, and a boolean feature to detect when month and day were swapped. Finally, we used the raw birth years as a feature for age. Figure 3.4 shows the features for date of birth.

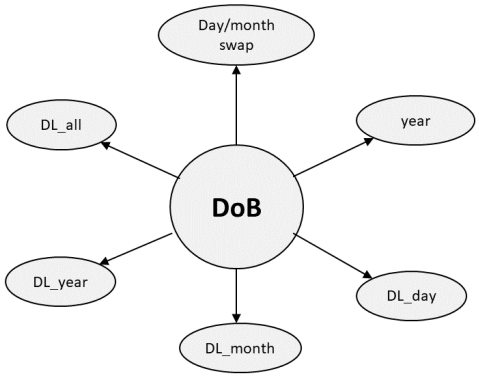

Figure 3.4: Date of birth features

3.2.2.3 Other Features

Besides the features mentioned before, For each pair of phone numbers, addresses, and social security numbers, we calculated the Damerau-Levenshtein and the Longest Common subsequence distance. We also engineered a binary feature to capture possible last name change due to marriage for females over eighteen because this is one of the common issues in RL in the USA. Finally, we coded gender as three categories based on the gender of the pair as ff, mm, or different. Figure 3.5 shows these features.

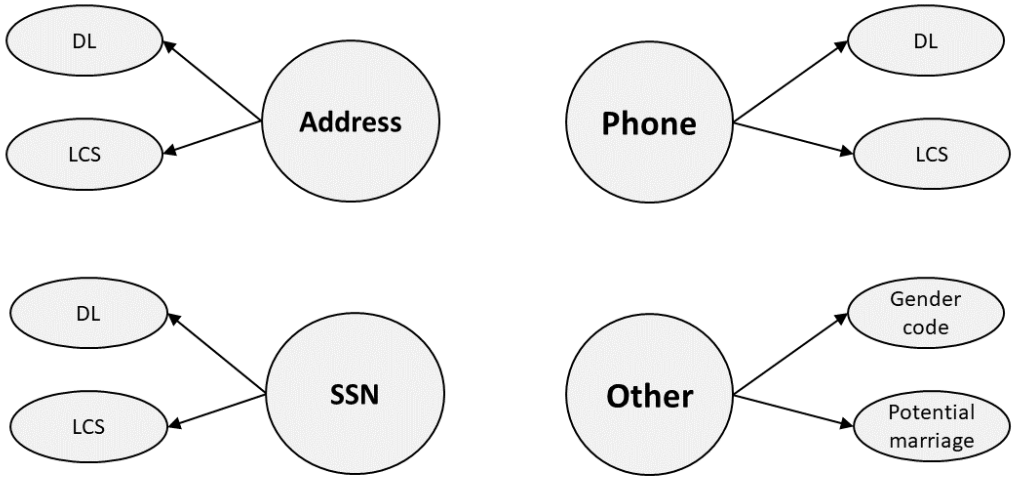

Figure 3.5: Other features

3.3 Machine Learning algorithms

After extracting the features, the next step is to train the ML algorithms using those features. First, we keep a part of training dataset as a validation set and train multiple models with different parameters. Then we tune hyper-parameters using the validation set. After finding the best hyper-parameters, we train our model one more time on the entire training dataset. This gives us the final model that can be used for testing. (Figure 3.6)

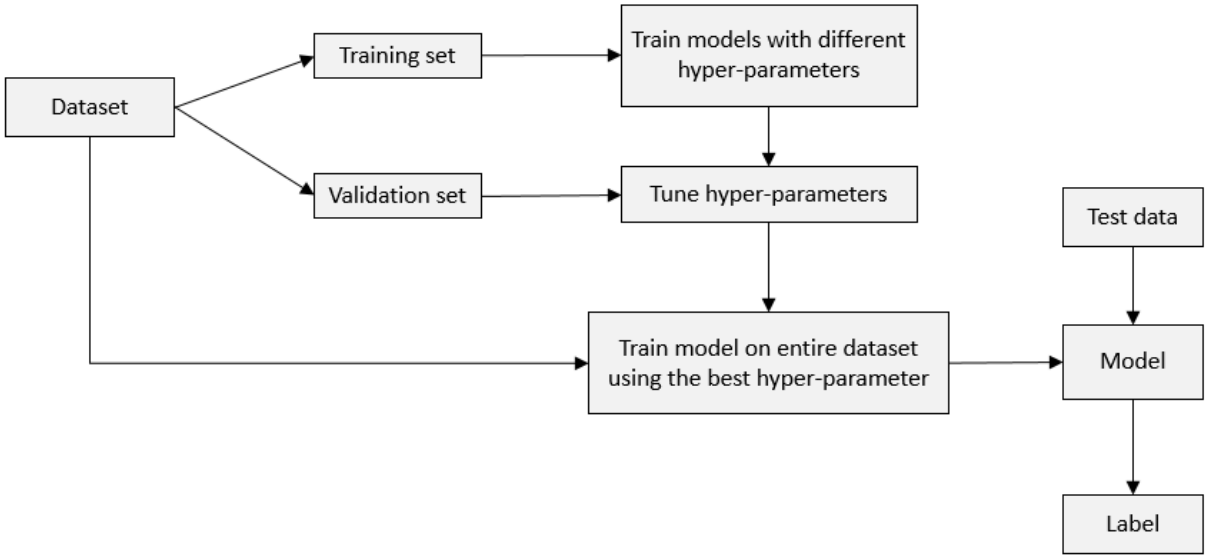

Figure 3.6: Machine Learning algorithm.

For this study, we have used four ML algorithms to generate different models: Random Forest, Linear and Radial Support Vector Machine, and Dense Neural Network.

3.3.1 Random Forest

The Random Forest classifier[25] is an ensemble method that trains several decision trees in parallel and then aggregates the results. Once the model is trained, each decision tree gives a label for the new instance. The predicted classes from all the trees are recorded and the class which is predicted by the maximum number of trees is considered as the final decision. RF able for classifying large data with accuracy. Figure 3.7 shows how RF works.

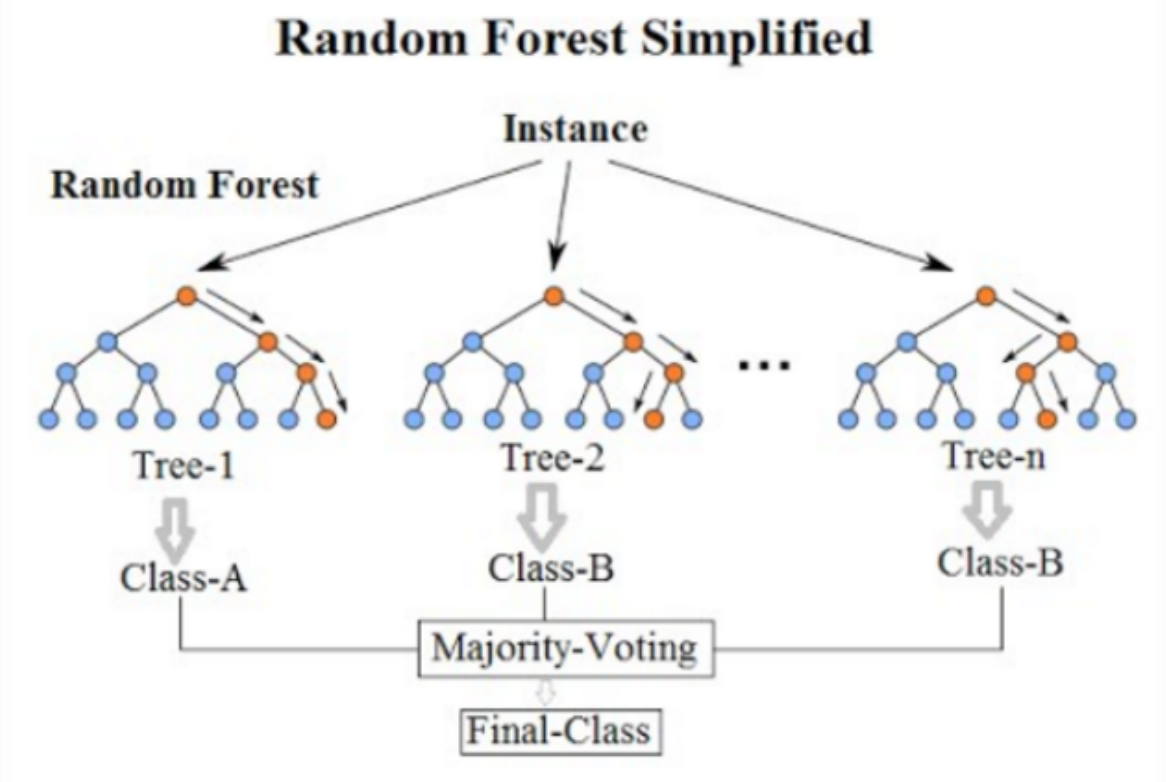

Figure 3.7: The Random Forest Classifier.

In order to train a random forest model, a grid search with 10-fold cross-validation was used on the training set to tune the maximum number of features at each split hyper-parameter, tested for 3, 5, 7, 9, 11, 13, and 15. As the number of estimators goes up, the performance typically goes up initially and plateaus after a point. The performance of the random forest started plateauing at about 250 estimators. Thus, the number of estimators was fixed at 350, which allowed a margin of 100 to ensure optimal performance. Once the best performing hyper-parameter was identified, the random forest model was rebuilt on all of the training data using the same hyper-parameter.

3.3.2 Radial SVM and Linear SVM

Support vector machines[26] are a set of supervised learning methods used for classification, regression, and outliers detection. SVMs are very effective in high dimensional spaces and can produce significant accuracy with less computation power. The objective of the support vector

machine algorithm is to find a hyperplane with maximum margin (see Figure 3.8), i.e the maximum distance between data points of all classes. Maximizing the margin distance causes more confident decisions for future data points.

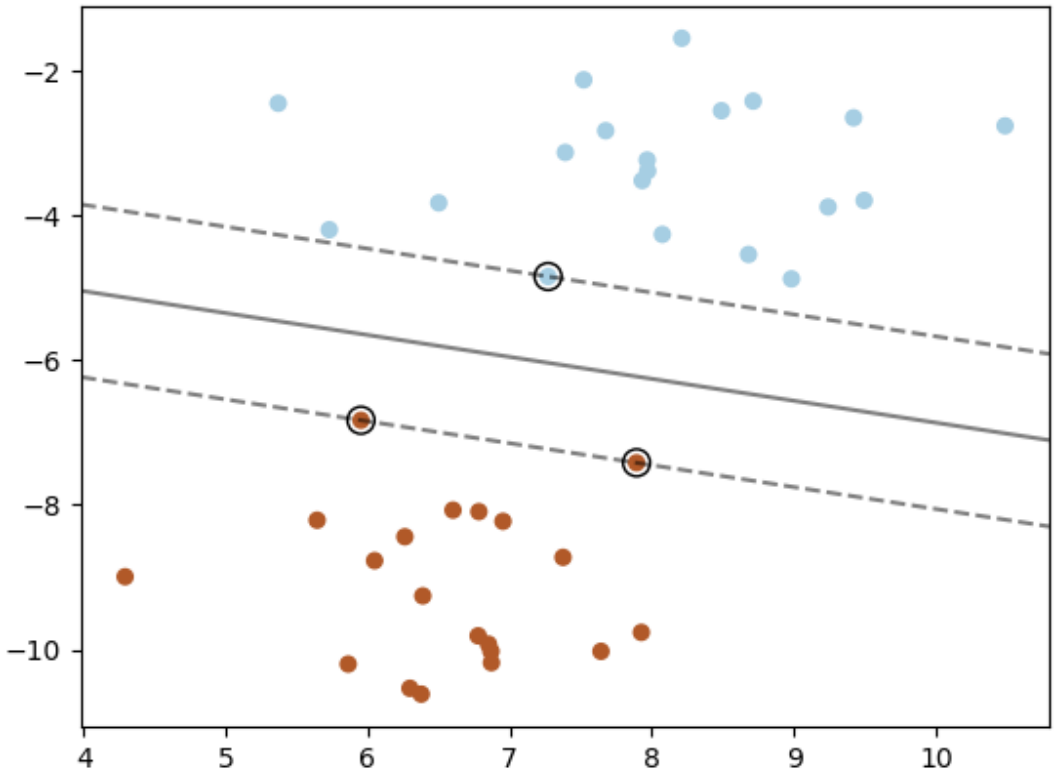

Figure 3.8: The SVM Classifier.

Two support vector machines were built, one with a radial basis function and the other with a linear basis function. The two key parameters for a radial basis kernel SVM were the penalty parameter, C and the kernel coefficient, sigma. Those two parameters were tuned using 10 fold cross-validation and grid search on the training data. The grid was validated for all combinations of C (0.1, 0.5, 1, 10) and sigma (0.03, 0.5, 0.9). The penalty parameter, C, was trained similarly for

the linear SVM. The models were retrained on all of the training data once the hyper-parameters were fixed.

3.3.3 Dense Neural Network

The last classifier we use is Dense Neural Network(DNN) which consists of simple processing nodes that are densely interconnected (Figure 3.9). Most of today's neural nets are organized into layers of nodes, and they're "feed-forward," meaning that data moves through them in only one direction.

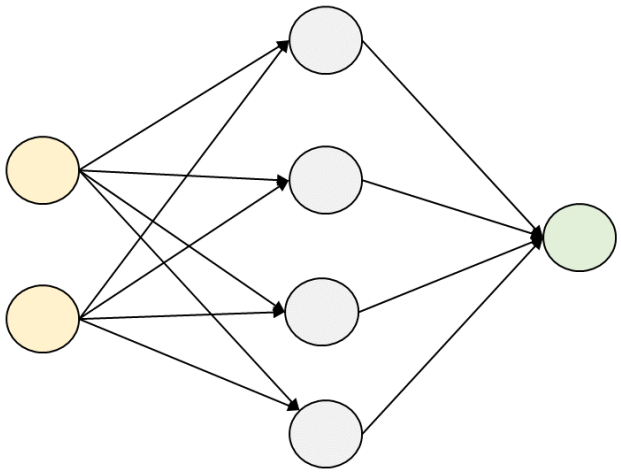

Figure 3.9: Dense Neural Network.

The neural net we used (Figure 3.10) had one input layer, two hidden layers, and one output layer with dense connections between all layers. The input layer had 33 units (from the feature vector discussed in section 3.2). The two hidden layers had 64 units each with relu activation functions (see Figure 3.11). After the first layer, there was batch normalization and 0.1% dropout. There was a batch normalization after the second layer but no dropout as is the standard practice for layers preceding the output layer. The output layer had 1 unit with a sigmoid activation(see

Figure 3.12) that returned the probability of a match. An RMSprop optimizer with a learning rate of 0.001 was used with binary cross-entropy as the loss function. The batch-size was maintained at the default 32 and the network was trained for 20 epochs. 20% of the data provided was used as the validation set for monitoring the validation performance.

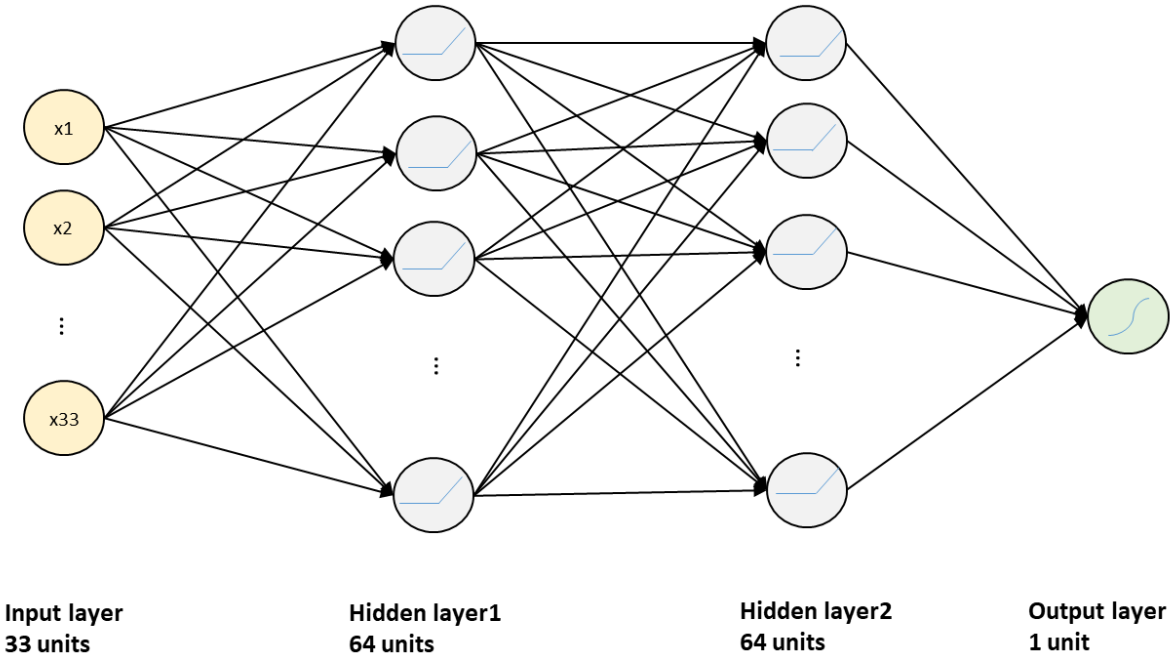

Figure 3.10: DNN model.

Activation function: ReLU $f(x) = x^+ = \max(0, x)$

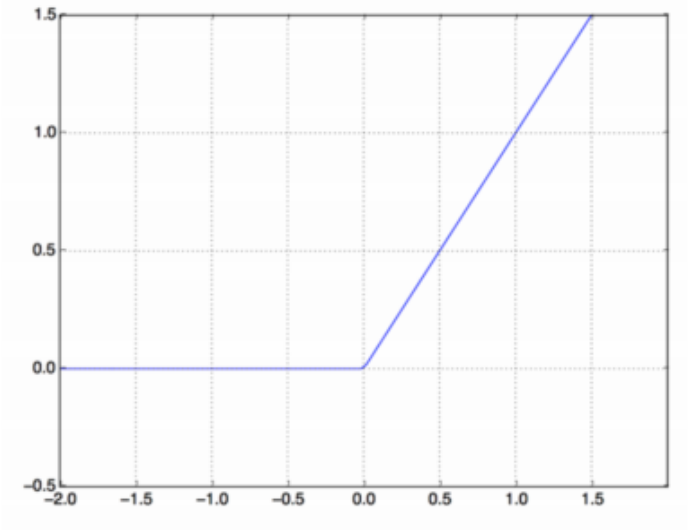

Figure 3.11: Relu activation function.

Activation function: sigmoid $S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$

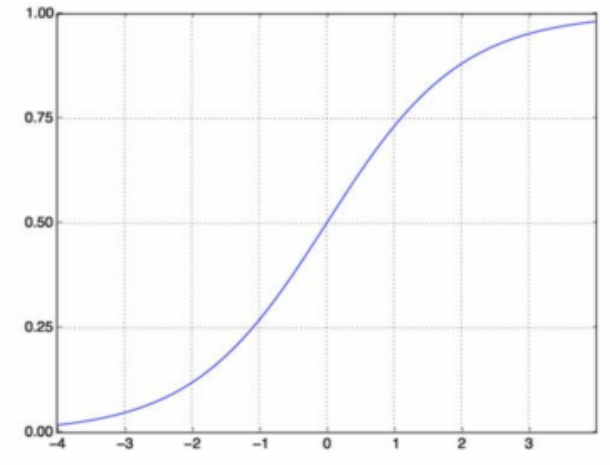

Figure 3.12: Sigmoid activation function.

4. EXPERIMENTAL DESIGN AND EVALUATION *

4.1 Data

In this study, we use two different real world datasets. The first is a large gold standard RL dataset to train the ML models[27], then we evaluate how well the models transfer to another dataset.

4.1.1 Hospital EHR data

For training our models, we used a large academic hospital EHR data. This dataset is based on 10,000,000 pairs that were generated from a hospital EHR dataset by blocking on first name and last name, first name and date of birth, last name and date of birth, and social security number. A gold standard dataset[27] was developed by randomly selecting 20,000 pairs and then reviewed by consensus among a team of 4 people through independent review. There are 31200 unique records in the gold standard data. Figure 4.1, demonstrate the gender distribution of this dataset. 52.3% of the records were female, 47.6% male, and the remaining 0.1% was undesignated. Age distribution is shown in Figure 4.2.

*Part of this chapter is reprinted from "Evaluation of Machine Learning Algorithms in a Human-Computer Hybrid Record Linkage System" by M. Ramezani, G. Ilangovan, H-C Kum 2021, AAAI-MAKE Symposium.

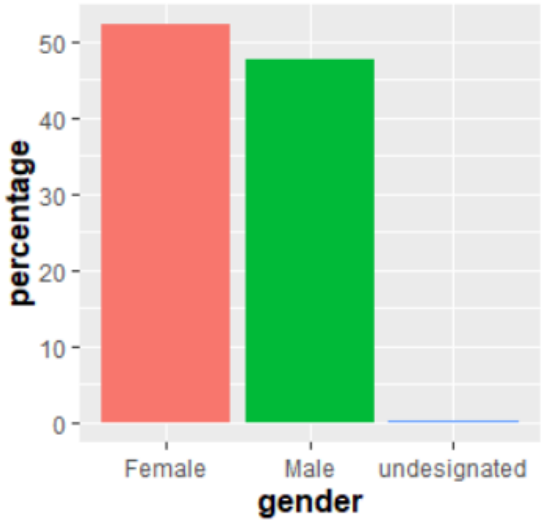

Figure 4.1: EHR data, gender distribution

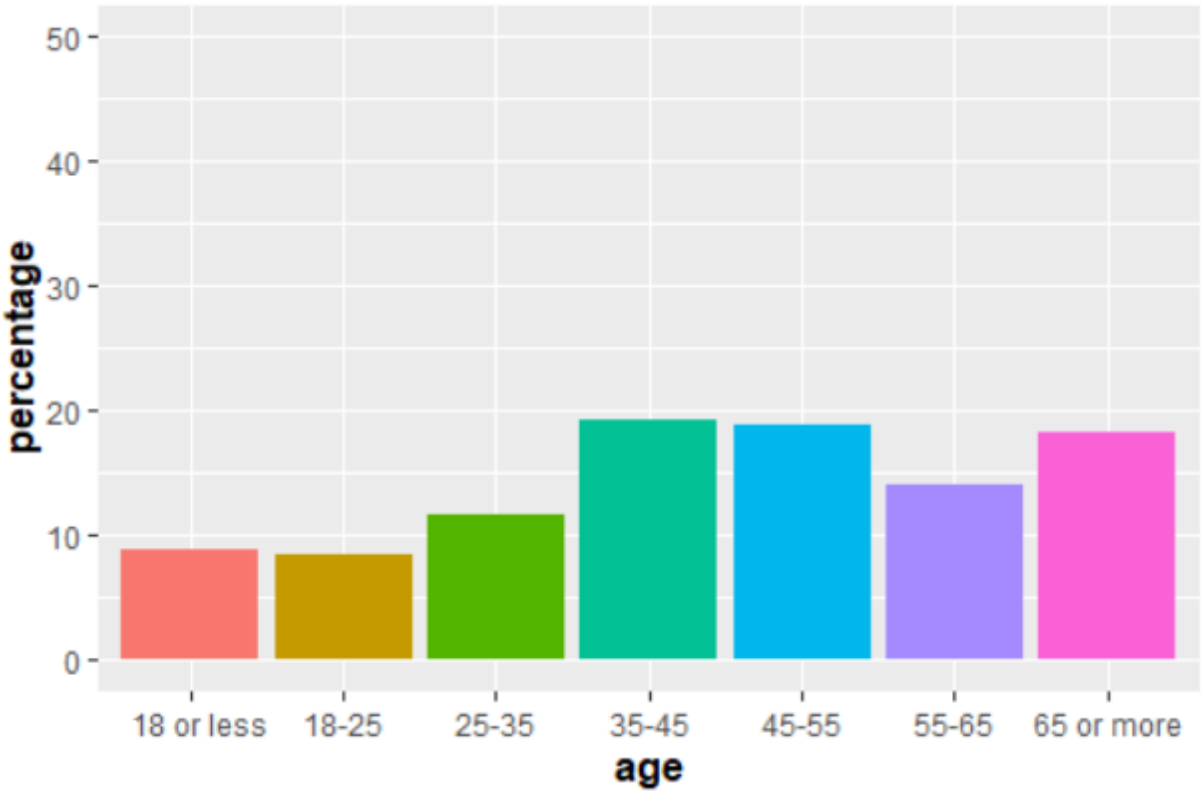

Figure 4.2: EHR data, age distribution

In our study, we randomly split the gold standard data into 10,000 for training data and 10,000 for test data. The test data had a total of 613 linkages that needed to be identified. The fields that were used directly for training the ML based RL models were:

- first name
- last name
- date of birth
- gender
- social security number
- phone
- address

4.1.2 NC voter registry data

The North Carolina State Board of Elections curates large amounts of data on state elections and voter registration. With several exceptions, this data is public and can be downloaded from `https://www.ncsbe.gov/results-data/voter-registration-data`. We link data from two time points (May 2017 and July 2020) using the voter registry number as the gold standard. Note that this dataset is only used to test the trained models. We perturb this data by randomly generating month and day of birth to account for twins as in [1]. Using the records from Yancey county, we generated 10,000 pairs by blocking on first name and last name, first name and date of birth, and last name and date of birth. There were a total of 1796 linkages that needed to be identified. There were 12700 unique records from which 52.8% of the records were female, 46.1% male, and the remaining 1.1% was undesignated. Since the data was from a voter registry, the minimum age on the month of data pull was 18 years. People with age 65 or above formed the biggest chunk of the records followed by middle aged populations (45 to 65). Figure 4.3, 4.4 demonstrate the gender and age distribution of data.

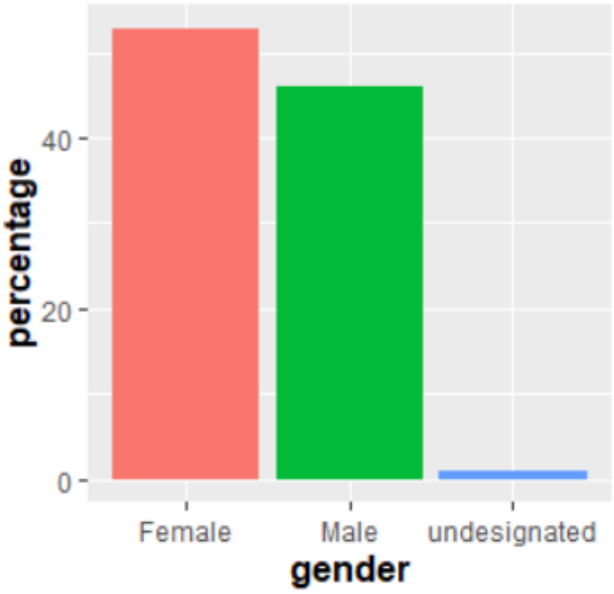

Figure 4.3: NC data, gender distribution

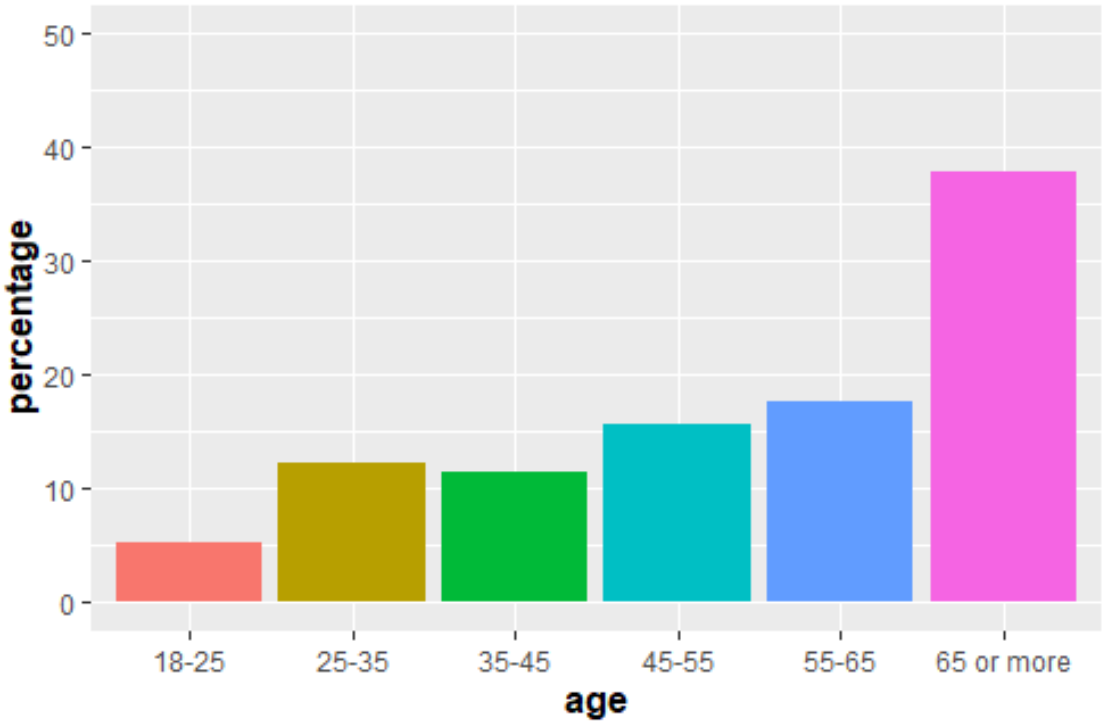

Figure 4.4: NC data, age distribution

4.2 Evaluation criteria

For evaluating the models we used three measures:

1. the number of pairs that need manual review
2. F1-score for automated results only
3. Recall over all results

F1-score is a common measure of linkage quality in automatic RL systems. It is the harmonic mean between precision and recall and hence is very useful in evaluating the effectiveness of linkage balancing between false positives and missing true links. Consider the following confusion matrix:

Actual Predicted	Match	Unmatch
Match	TP	FN
Unmatch	FP	TN

where TP is the True Positive, TN is the True Negative, FP is the False Positive and FN is the False Negative. Then:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.3}$$

In the hybrid RL system, the number of pairs that need manual review is determined by two thresholds (Figure 4.5), T1 and T2, that are used to determine uncertain pairs. In this study, we defined T1 and T2 in terms of Positive Predictive Value (PPV) and Negative Predictive Value (NPV) which are calculated as:

$$\begin{aligned} Positive\,Predictive\,Value\,(PPV) &= \frac{number\,of\,true\,positives}{number\,of\,positive\,calls} \\ &= \frac{TP}{TP+FP} \end{aligned} \tag{4.4}$$

$$\begin{aligned} Negative\,Predictive\,Value\,(NPV) &= \frac{number\,of\,true\,negatives}{number\,of\,negative\,calls} \\ &= \frac{TN}{TN+FN} \end{aligned} \tag{4.5}$$

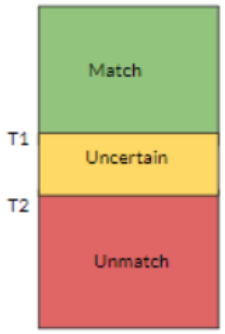

Figure 4.5: Two thresholds are needed for hybrid RL.

Since accurate results are very important in the health domain, we selected T1 and T2 such that among all predictions with probabilities above and below them respectively, the predictions were perfect on the training data ($PPV = NPV = 1$). We use the F1-score for ML RL models ($F1_{autoRL}$), which focuses on the effectiveness of only the subset of pairs that are labeled using the automated methods outside T1 and T2.

In addition, we use $Recall_{overall}$, which takes into account the full set of pairs of the whole hybrid system and depicts how much of the overall TP have been correctly identified by the automated methods. If the $Recall_{overall}$ is too low, that means the automated methods are mainly good at reducing the manual review work and rely mostly on the manual process to detect much of the correct links and is roughly correlated with the manual review set size. On the other hand, very high $Recall_{overall}$ shows that the ML models captured most of the linkage and we may not need to spend much time for manual review.

4.2.1 Study design

Three experiments have been designed to answer three study questions:

1. How well do the four ML RL algorithms meet the two goals of the hybrid RL system?
2. How well do the four ML RL models trained on one dataset transfer to different settings?
3. How does adding the n2v feature affect the different results in the experiments?

In the first experiment, we used only the hospital gold standard data (EHR data). We trained using the 10,000 pairs in the training dataset then evaluated the performance on the other 10,000 testing data. For each pair, we calculated the 33 features described in section 3.2 (excluding the two n2v features). In the training phase, 9000 pairs were used for training and 1000 pairs were used as the validation set to tune the hyper-parameters. After hyper-parameter tuning, we trained the models with the selected hyper-parameter one more time on the whole training set. Finally, we test the trained models on the test set and record the results.

In our second experiment, we test the trained models from the previous experiment on 10,000 pairs randomly generated from the NC voter dataset to see how well different ML models transfer to different settings. We ran this experiment 100 times and report the mean and standard deviation for each model (Table 5.3).

Finally, as the last experiment, the first and the second experiments were repeated but this time the two n2v features, one feature for first name and one for last name, were added. Thus, in this

experiment, each record had 35 features. The main purpose of this experiment was to observe the effectiveness of the n2v distance on the hybrid record linkage process.

5. RESULTS*

5.1 The performance of different ML models

Table 5.1 compares the four ML algorithms. Both training and test datasets are from the hospital data (EHR data). The results of this experiment demonstrate that although $F1_{autoRL}$ scores were comparable across all methods, random forest and linear SVM did worst in the $Recall_{overall}$ at slightly over 60%. Close to 40% of the linkages has to be found through manual review. This is consistent with the considerably bigger manual review size for these two algorithms compared to Radial SVM and DNN.

Model	manual review	$Recall_{overall}$	$F1_{autoRL}$	TP	FP	TN	FN	Total
RF	306	0.625	0.992	383	1	9305	5	9694
Radial SVM	187	0.721	0.98	442	2	9353	16	9813
Linear SVM	321	0.612	0.991	375	0	9297	7	9679
DNN	217	0.936	0.989	574	11	9196	2	9783

Table 5.1: Experiment 1. Comparing the performance of four ML algorithm for RL on EHR data (There are 613 linkages).. Reprinted from [2]

5.2 The performance of different ML models on a new setting

Experiment 2 studied how well the ML models trained in one setting transfers to different data. We used the trained model that we created using the EHR data (experiment 1) to test those models on a different dataset (Voter Registry data). As seen in Table 5.2, the two SVM models transfer to the new setting best, followed by RF, then DNN. Although the manual review set size are bigger

*Part of this chapter is reprinted from "Evaluation of Machine Learning Algorithms in a Human-Computer Hybrid Record Linkage System" by M. Ramezani, G. Ilangovan, H-C Kum 2021, AAAI-MAKE Symposium. © 2021 Copyright for this paper by its authors.

than previous experiments, the $F1_{autoRL}$ is still reasonable in the three models which is important because the manual phase can then fill in the gap even if $Recall_{overall}$ is somewhat low. DNN model is not usable because there are too many FP and FN, that cannot be overcome in the manual review phase.

Model	manual review	$Recall_{overall}$	$F1_{autoRL}$	TP	FP	TN	FN	Total
RF	4479	0.197	0.987	354	1	5158	8	5521
Radial SVM	1857	0.815	0.986	1463	7	6637	36	8143
Linear SVM	1553	0.589	0.991	1057	1	7371	18	8447
DNN	1836	0.175	0.198	315	1393	5295	1161	8164

Table 5.2: Experiment 2. Transfering trained ML RL models to voter data (1773 linkages). Reprinted from [2]

We repeated this experiment 100 times and record the results. Table 5.3 shows the mean and standard deviation of the 100 repeated experiments using voter data.

Model	manual review size	$Recall_{overall}$	$F1_{autoRL}$
RF	4459 ± 43.5	0.193 ± 0.00768	0.982 ± 0.00443
Radial SVM	1841.5 ± 32.5	0.81 ± 0.00809	0.983 ± 0.00192
Linear SVM	1546.9 ± 29.2	0.58 ± 0.00968	0.989 ± 0.00181
DNN	1827.6 ± 29.6	0.171 ± 0.00822	0.194 ± 0.00918

Table 5.3: Experiment 2. The mean and standard variation of 100 runs on voter data. Reprinted from [2]

5.2.1 Effect of n2v feature on model performance

For the last experiment, we added 2 more features: n2v distance for first name and last name to see how adding these features can change the results in the first and the second experiments. As seen in Table 5.4, adding n2v caused a big reduction in the number of pairs that needed manual review, significant improvements to $Recall_{overall}$, with only slight change in $F1_{autoRL}$ for all models except DNN. Figures 5.1, 5.2, and 5.3 depicts the effect of adding n2v features to ML model performance in the first experiment.

Model	manual review	$Recall_{overall}$	$F1_{autoRL}$	TP	FP	TN	FN	Total
RF + n2v	75	0.962	0.985	590	15	9317	3	9925
Radial SVM + n2v	34	0.956	0.975	586	16	9350	14	9966
Linear SVM + n2v	64	0.967	0.977	593	24	9315	4	9936
DNN + n2v	239	0.93	0.99	570	10	9180	1	9761

Table 5.4: Experiment 3. The effect of adding n2v on the performance of ML algorithms: EHR data (613 linkages). Reprinted from [2]

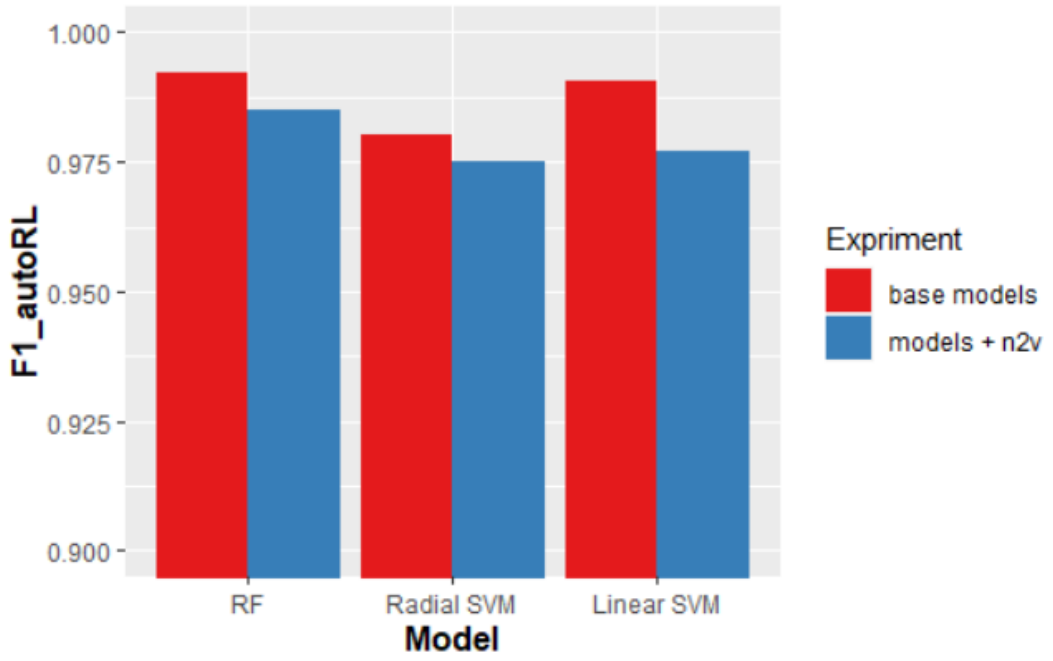

Figure 5.1: Experiment 3. The effect of adding n2v on F1-score: EHR data (613 linkages). Reprinted from [2]

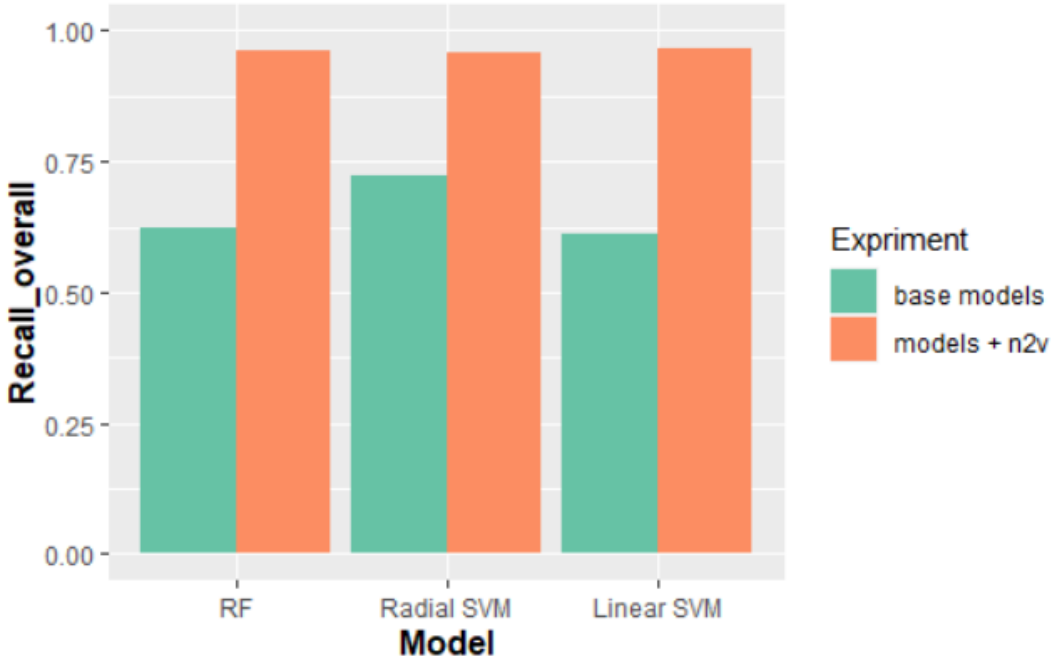

Figure 5.2: Experiment 3. The effect of adding n2v on recall: EHR data (613 linkages). Reprinted from [2]

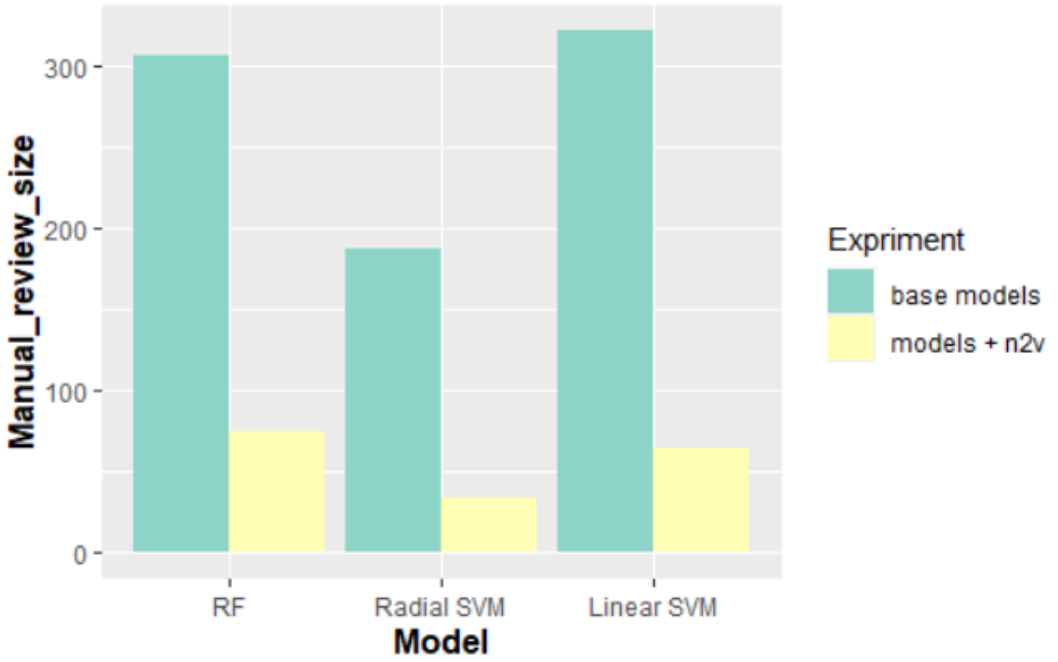

Figure 5.3: Experiment 3. The effect of adding n2v on manual review size: EHR data (613 linkages). Reprinted from [2]

Results were somewhat similar for the second experiment presented in Table 5.5, although the impact on $Recall_{overall}$ was not seen. It seems that adding n2v, increases the confidence (higher probabilities) of the model predictions which means potentially more matches and unmatches are detected through automated step, and the number of observations classified as uncertain is smaller. However in the DNN model, the impact was opposite with higher manual review size and slightly higher F1-scores. Figures 5.4, 5.5, and 5.6 shows the effect of adding n2v features to ML model performance in the second experiment (voter data).

Model	manual review	$Recall_{overall}$	$F1_{autoRL}$	TP	FP	TN	FN	Total
RF + n2v	4386	0.215	0.988	386	1	5219	8	5614
Radial SVM + n2v	927	0.773	0.983	1388	2	7638	45	9073
Linear SVM + n2v	1689	0.33	0.98	594	0	7693	24	8311
DNN + n2v	3015	0.172	0.213	309	1329	4391	956	6985

Table 5.5: Experiment 3. The effect of adding n2v on the performance of ML algorithms: Voter data (1773 linkages). Reprinted from [2]

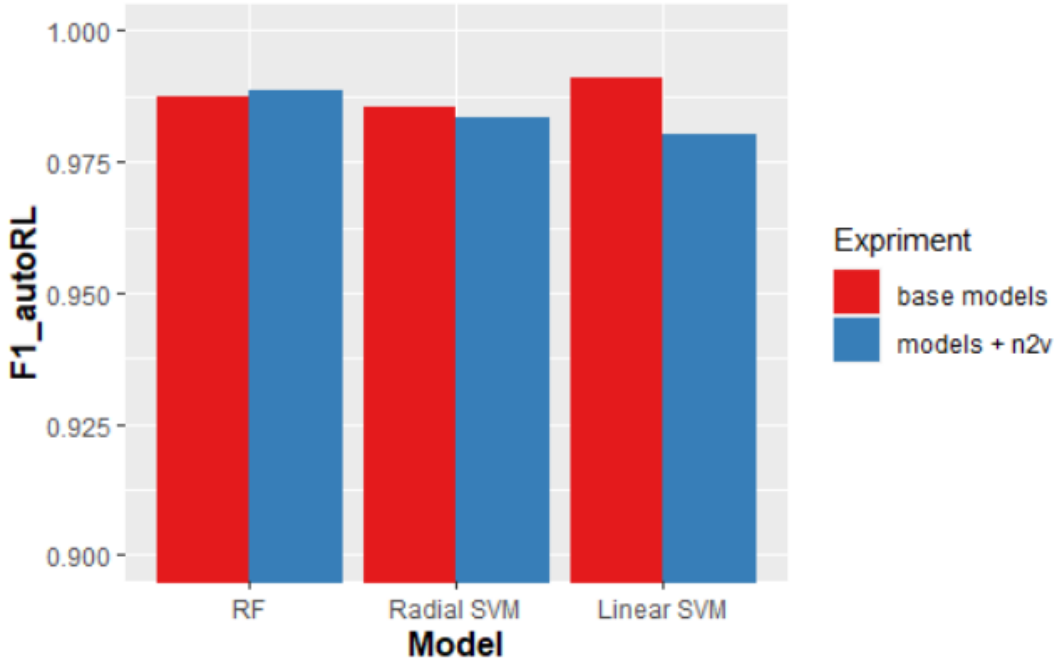

Figure 5.4: Experiment 3. The effect of adding n2v on F1-score: voter data (1773 linkages). Reprinted from [2]

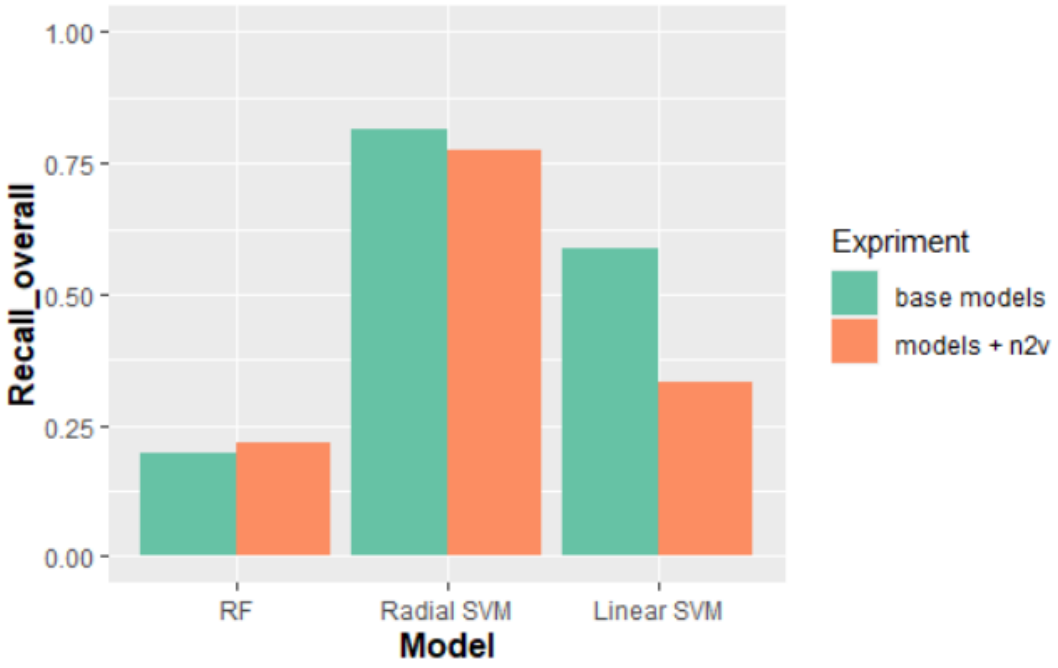

Figure 5.5: Experiment 3. The effect of adding n2v on recall: voter data (1773 linkages). Reprinted from [2]

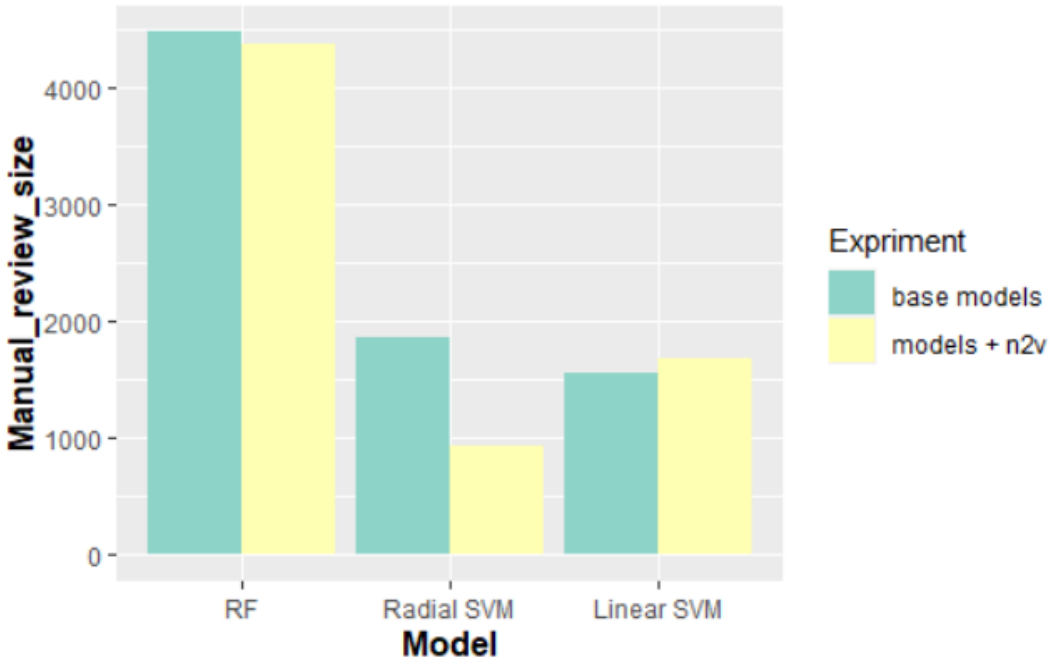

Figure 5.6: Experiment 3. The effect of adding n2v on manual review size: voter data (1773 linkages). Reprinted from [2]

Similar to the second experiment, we repeated this experiment 100 times. Table 5.6 shows the mean and standard deviation of the 100 repeated experiments using voter data.

Model	manual review size	$Recall_{overall}$	$F1_{autoRL}$
RF + n2v	4378.4 ± 41.5	0.212 ± 0.00789	0.986 ± 0.0032
Radial SVM + n2v	922.8 ± 27.2	0.769 ± 0.0088	0.983 ± 0.00196
Linear SVM + n2v	1662.1 ± 33.5	0.334 ± 0.00898	0.976 ± 0.0035
DNN + n2v	2997 ± 38.9	0.163 ± 0.00831	0.203 ± 0.00997

Table 5.6: Experiment 3. The mean and standard variation of 100 runs on voter data. Reprinted from [2]

6. DISCUSSION *

This research sought to systematically study the performance of the different ML algorithms (Random Forest, Linear SVM, Radial SVM, and Dense Neural Network) on different settings for a hybrid record linkage method in terms of F1 score, Recall, and size of manual review. The automatic ML based RL code and models can be downloaded from `https://github.com/pinformatics/hybridRL_code_and_models`. Users can use the trained models to conduct record linkage on their data or train a new model using their own dataset.

In this study, we designed and ran three experiments. The first experiment was designed to compare four ML algorithms and the most interesting finding was that although RF had the best $F1_{autorRL}$ score, it was not a good model overall for the hybrid system. The manual review size for radial SVM was only 61% of random forest model at the cost of having more false labels, but little impact of $F1_{autoRL}$ scores. Radial SVM missed 16 true matches while DNN had 11 false links. And both these models were able to identify many more of the linkages, giving much better $Recall_{overall}$ scores.

It is very clear that in a hybrid system the price for perfect performance in the first pass has to be paid by a lot of manual review in the second pass. Obviously, the performance requirements for the algorithms are affected by the importance of the linkage task. When medical databases are to be linked[27], the process is often critical and requires thresholds that give perfect (100%) results in the training set. However, in some domains like genealogy records, small error rates are often acceptable. In that case, users can relax the thresholds to meet project requirements that can result in smaller manual review set. Thus, depending on the performance requirements of the task at hand, the performance requirements for the automatic linkage can be defined. This can potentially save a lot of time and effort on the manual linkage.

As a second experiment, we studied how well different ML based RL models transfer to a

*Part of this chapter is reprinted from "Evaluation of Machine Learning Algorithms in a Human-Computer Hybrid Record Linkage System" by M. Ramezani, G. Ilangovan, H-C Kum 2021, AAAI-MAKE Symposium. © 2021 Copyright for this paper by its authors.

different setting. The results of this experiment demonstrate that RF, linear SVM and radial SVM models transfer to a new setting much better than DNN. For these three models, $F1_{autoRL}$ score was comparable, however the manual review size is much bigger than previous experiments and impact the $Recall_{overall}$ score. The results indicate that the models built on EHR data can be used to identify clear non-matches, and identify some number of true matches but manual review is required to identify most of the true matches. SVM models perform much better than the RF by reducing the manual review size to less than 42% while also identifying over 60% of the linkages. In comparison, RF model had over 40% manual review size and only 20% of linkages. DNN performance was not acceptable to be used and seems to indicate that the model may be over fitting to the data it was trained on.

The goal of the last experiment was to see how adding the n2v features, which is a letter embedding for names, can affect the results. Clearly adding n2v features can reduce the size of manual review significantly on all models except DNN in the first experiment (Table 5.4). As expected, the results indicate that using n2v distance for first name and last name is similar to the impact of adding in approximate name matching where it can increase identification of true linkages automatically but at the cost of also increasing false linkages. More concretely, $Recall_{overall}$ improved noticeably to over 90% when n2v features were added, but at the same time number of FP went up from 1 ,2 ,0 to 15, 16, 24 respectively for RF, Radial SVM, and Linear SVM. Remember that these errors cannot be corrected during the manual review phase, so the choice of using n2v or not will depend on the error rate that is acceptable in the given application. Adding n2v features had little impact on the DNN model performance which was low anyway.

The impact of adding n2v features to model performance applied to a different dataset (voter data) in Table 5.5 was seen most in the two SVM models. Both RF and DNN had comparable low results on all measures except DNN that has increased the manual review size making things worse. In comparison, radial SVM models reduced the manual review size by more than half (1857 to 927). The interesting finding was that this reduction did not translate directly into improvements in $Recall_{overall}$ where radial SVM had a significant reduction to 77%. Upon closer look, we can

see that most of the reduction in manual review was due to pairs that were confirmed correctly as TN in the radial SVM model. In comparison, the linear SVM dropped many TP (from 1057 to 594) when n2v was added reducing $Recall_{overall}$. Thus, the radial SVM model benefited the most from adding in the n2v features with negligible impact on Recall and F1 score.

7. CONCLUSIONS

Automatic record linkage methods have made significant progress during the last few decades, however they still may not have the high degree of reliability of manual record linkage. On the other hand, the manual record linkage method is very expensive and time-consuming. Thus, in this research, we presented and evaluated an open source hybrid record linkage framework that combines the manual process and the automated process to achieve both scalability and high quality linkage results.

Using this framework, we first trained models based on four popular machine learning algorithms (Random Forest, Linear SVM, Radial SVM, and Dense Neural Network) and compared their performance in terms of F1-score, recall and the size of manual review set. In our second experiment, we studied how different trained models transfer to a new setting. The results showed that RF, linear SVM and radial SVM models transfer to a new setting much better than DNN. Finally, in our last experiment, we studied the effect of name2vec, as two features to measure the distance between first names and last names in pairs separately, on the models' performance. The results show that using name2vec caused the models' predictions with higher probabilities which left fewer pairs in the uncertain class. Overall the SVM models performed best in all experiments.

More work is needed to test if more complex and effective neural network models may have better performance. In addition, future work is needed to systematically quantify the biases in RL by race to study the impact of RL on health disparities database studies.

REFERENCES

[1] G. Ilangovan, "Benchmarking the effectiveness and efficiency of machine learning algorithms for record linkage," Master's thesis, 2019.

[2] M. Ramezani, G. Ilangovan, and H.-C. Kum, "Evaluation of machine learning algorithms in a human-computer hybrid record linkage system," 2021.

[3] E. L. e. a. Fosbøl, "Prehospital system delay in st-segment elevation myocardial infarction care: A novel linkage of emergency medicine services and inhospital registry data," *American Heart Journal*, vol. 165, no. 3, pp. 363–370, 2013.

[4] H. B. Newcombe, J. M. Kennedy, S. Axford, and A. P. James, "Automatic linkage of vital records," *Science*, vol. 130, no. 3381, pp. 954–959, 1959.

[5] M. Karim, M. Ramezani, T. Sunbury, R. Ohsfeldt, and H.-C. Kum, "View: a framework for organization level interactive record linkage to support reproducible data science," *arXiv preprint arXiv:2102.08273*, 2021.

[6] H.-C. Kum, E. D. Ragan, G. Ilangovan, M. Ramezani, Q. Li, and C. Schmit, "Enhancing privacy through an interactive on-demand incremental information disclosure interface: applying privacy-by-design to record linkage," in {*SOUPS*} *2019*, 2019.

[7] E. D. Ragan, H.-C. Kum, G. Ilangovan, and H. Wang, "Balancing privacy and information disclosure in interactive record linkage with visual masking," in *SGICHI 2018*, pp. 1–12, 2018.

[8] M. e. a. Bailey, "How well do automated linking methods perform in historical data? evidence from new us ground truth.," tech. rep., Mimeo, 2018.

[9] H.-C. Kum, A. Krishnamurthy, A. Machanavajjhala, M. K. Reiter, and S. Ahalt, "Privacy preserving interactive record linkage (ppirl)," *Journal of the American Medical Informatics Association*, vol. 21, no. 2, pp. 212–220, 2014.

[10] J. Foxcroft, A. d'Alessandro, and L. Antonie, "Name2vec: Personal names embeddings," in *Canadian Conference on Artificial Intelligence*, pp. 505–510, Springer, 2019.

[11] L. Antonie, K. Inwood, D. J. Lizotte, and J. A. Ross, "Tracking people over time in 19th century canada for longitudinal analysis," *Machine learning*, vol. 95, no. 1, pp. 129–146, 2014.

[12] K. Kim and C. L. Giles, "Financial entity record linkage with random forests," in *Proceedings of the Second International Workshop on Data Science for Macro-Modeling*, pp. 1–2, 2016.

[13] P. Treeratpituk and C. L. Giles, "Disambiguating authors in academic publications using random forests," in *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pp. 39–48, 2009.

[14] B. Pixton and C. Giraud-Carrier, "Using structured neural networks for record linkage," in *Proceedings of the Sixth Annual Workshop on Technology for Family History and Genealogical Research*, 2006.

[15] D. R. Wilson, "Beyond probabilistic record linkage: Using neural networks and complex features to improve genealogical record linkage," in *The 2011 international joint conference on neural networks*, pp. 9–14, IEEE, 2011.

[16] J. J. Feigenbaum, "Automated census record linking: A machine learning approach," 2016.

[17] P. Kaur *et al.*, "A comparison of machine learning classifiers for use on historical record linkage," Master's thesis, 2020.

[18] J. M. Bronstein, C. T. Lomatsch, D. Fletcher, T. Wooten, T. M. Lin, R. Nugent, and C. L. Lowery, "Issues and biases in matching medicaid pregnancy episodes to vital records data: the arkansas experience," *Maternal and child health journal*, vol. 13, no. 2, pp. 250–259, 2009.

[19] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989.

[20] W. E. Winkler and Y. Thibaudeau, *An application of the Fellegi-Sunter model of record linkage to the 1990 US decennial census*. Citeseer, 1991.

[21] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[22] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[23] M. K. Odell, "The profit in records management," *Systems (New York)*, vol. 20, p. 20, 1956.

[24] R. Russell and M. Odell, "The soundex indexing system," *National Archives and Records Administration*, 1918.

[25] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[27] E. Joffe, M. J. Byrne, P. Reeder, J. R. Herskovic, C. W. Johnson, A. B. McCoy, D. F. Sittig, and E. V. Bernstam, "A benchmark comparison of deterministic and probabilistic methods for defining manual review datasets in duplicate records reconciliation," *JAMIA*, vol. 21, no. 1, pp. 97–104, 2014.